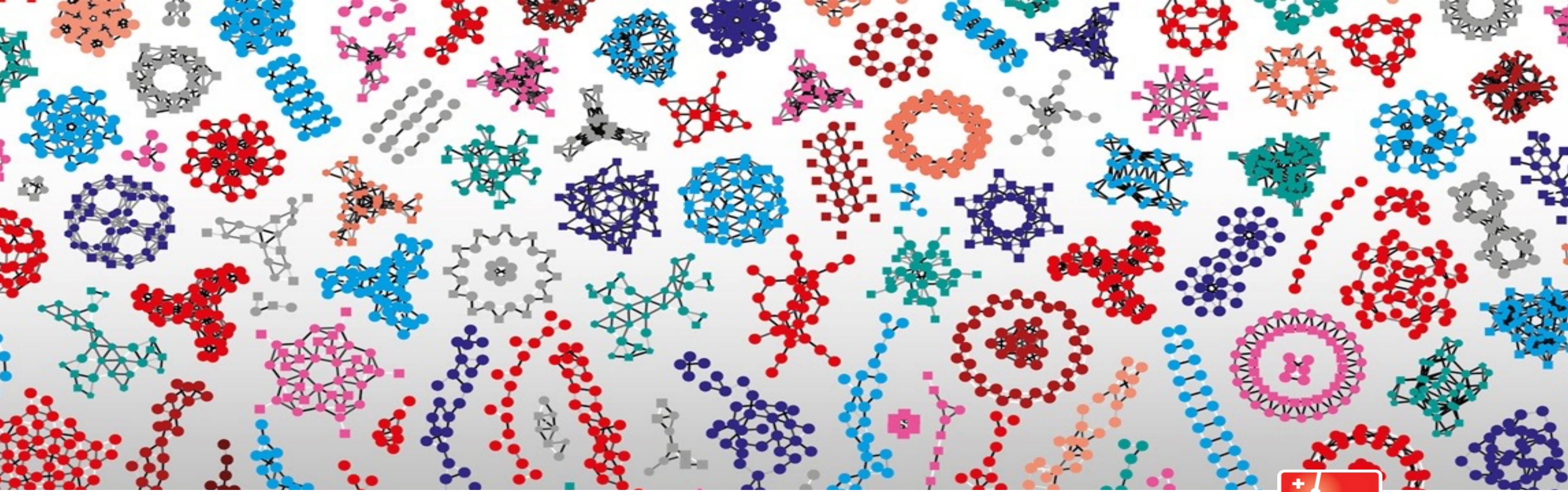


Swiss Institute of Bioinformatics

First Steps with UNIX in Life Sciences

Grégoire Rossier, Robin Engler

Learning outcomes

- To get familiar with the UNIX environment
- To use the most common UNIX commands
- To acquire skills necessary for further courses requiring UNIX, like HPC and NGS analysis courses.

Outline

1. What is UNIX and why should biologists use it?
2. UNIX filesystem: navigation and usage
3. Environment, processes & Jobs
4. Working with directories and files
5. Working with file content

Exercises are integrated in chapters

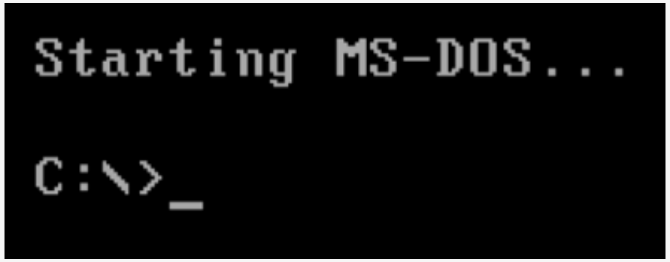
An optional exam will be put online at the end of the course

What is UNIX and why should biologists use it?

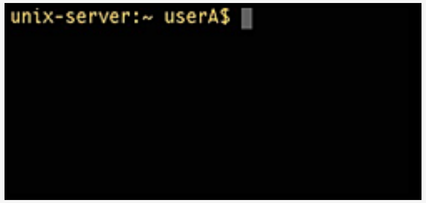
Swiss Institute of Bioinformatics

Command Line Interface (CLI)

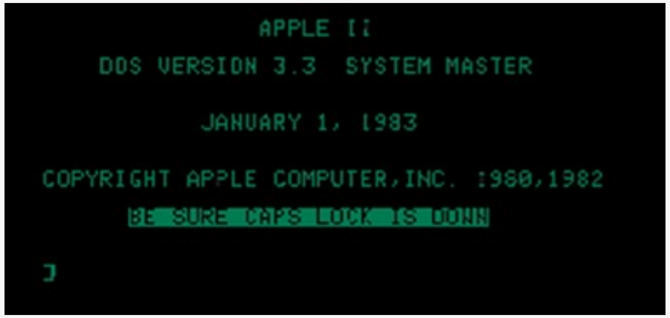
In the 70's, all systems used a command line interface



```
Starting MS-DOS...

C:\>_
```



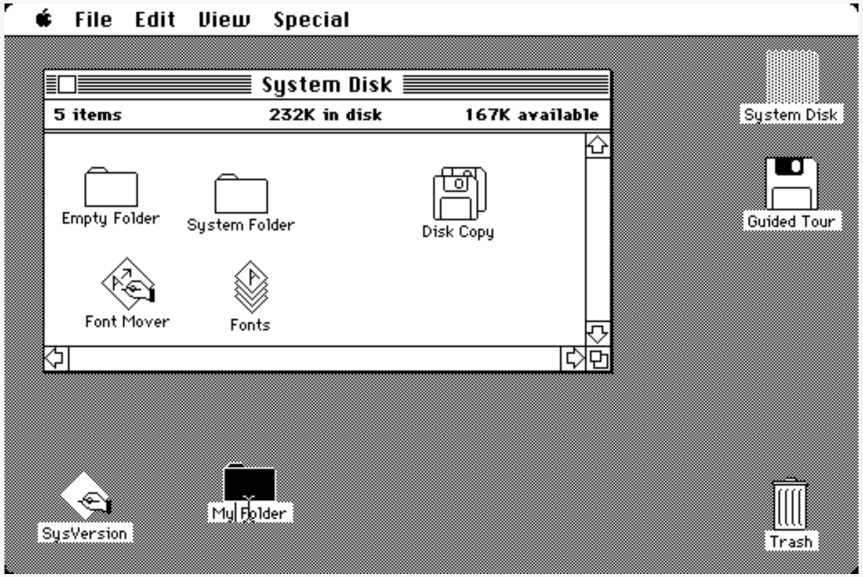
```
unix-server:~ userA$
```



```
                APPLE ][
      DOS VERSION 3.3   SYSTEM MASTER

            JANUARY 1, 1983

   COPYRIGHT APPLE COMPUTER,INC. 1980,1982
         BE SURE CAPS LOCK IS DOWN

]
```

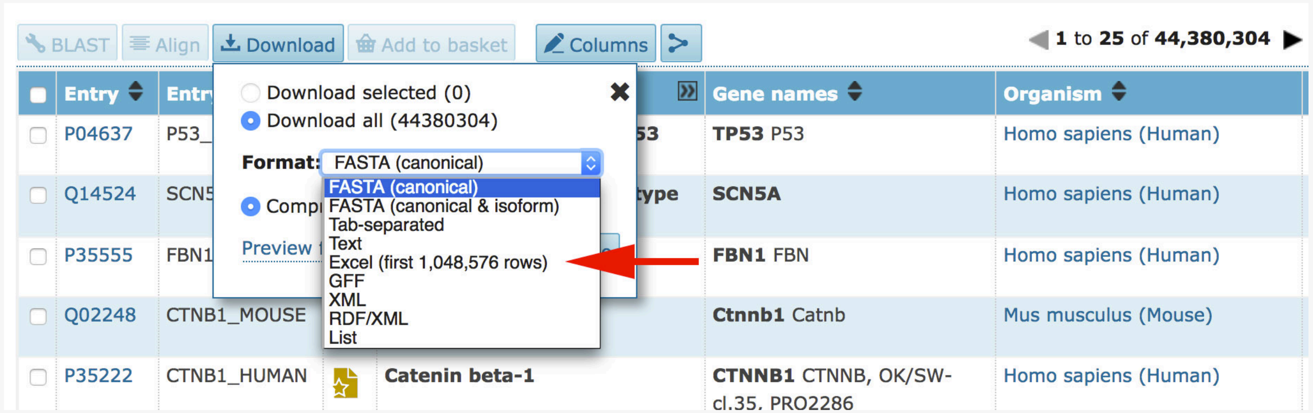
Graphical User Interface (GUI)

- Development since the 60's
- First popular product was the Macintosh in **1984**
- **Different way of thinking** / acting: windows, menus, icons, mouse and clicks (now touchscreens)



CLI vs GUI – size issues

Dealing with more than 1 million lines



BLAST | Align | Download | Add to basket | Columns

1 to 25 of 44,380,304

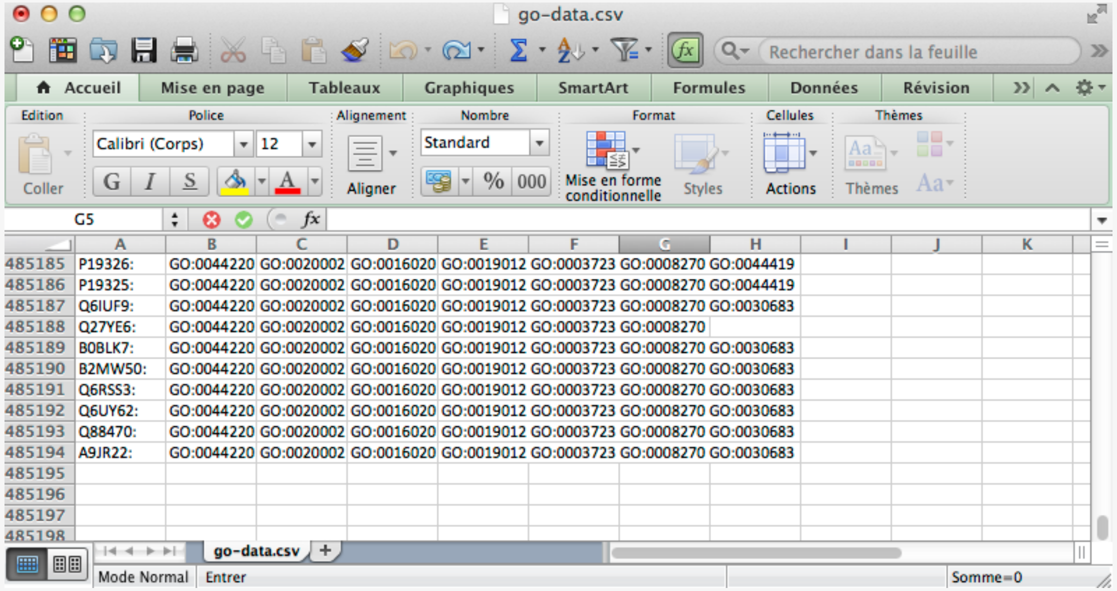
Download selected (0)
[✓ Download all (44380304)]

Format: FASTA (canonical)

FASTA (canonical)
FASTA (canonical & isoform)
Tab-separated
Text
Excel (first 1,048,576 rows)
GFF
XML
RDF/XML
List

Entry	Entry name	Protein names	Gene names	Organism
P04637	P53_		TP53 P53	Homo sapiens (Human)
Q14524	SCN5		SCN5A	Homo sapiens (Human)
P35555	FBN1		FBN1 FBN	Homo sapiens (Human)
Q02248	CTNB1_MOUSE		Ctnnb1 Catnb	Mus musculus (Mouse)
P35222	CTNB1_HUMAN	Catenin beta-1	CTNNB1 CTNNB, OK/SW-cl.35, PRO2286	Homo sapiens (Human)

CLI vs GUI - complex queries



	A	B	C	D	E	F	G	H	I	J	K
485185	P19326:	GO:0044220	GO:0020002	GO:0016020	GO:0019012	GO:0003723	GO:0008270	GO:0044419			
485186	P19325:	GO:0044220	GO:0020002	GO:0016020	GO:0019012	GO:0003723	GO:0008270	GO:0044419			
485187	Q6IUF9:	GO:0044220	GO:0020002	GO:0016020	GO:0019012	GO:0003723	GO:0008270	GO:0030683			
485188	Q27YE6:	GO:0044220	GO:0020002	GO:0016020	GO:0019012	GO:0003723	GO:0008270				
485189	B0BLK7:	GO:0044220	GO:0020002	GO:0016020	GO:0019012	GO:0003723	GO:0008270	GO:0030683			
485190	B2MW50:	GO:0044220	GO:0020002	GO:0016020	GO:0019012	GO:0003723	GO:0008270	GO:0030683			
485191	Q6RSS3:	GO:0044220	GO:0020002	GO:0016020	GO:0019012	GO:0003723	GO:0008270	GO:0030683			
485192	Q6UY62:	GO:0044220	GO:0020002	GO:0016020	GO:0019012	GO:0003723	GO:0008270	GO:0030683			
485193	Q88470:	GO:0044220	GO:0020002	GO:0016020	GO:0019012	GO:0003723	GO:0008270	GO:0030683			
485194	A9JR22:	GO:0044220	GO:0020002	GO:0016020	GO:0019012	GO:0003723	GO:0008270	GO:0030683			
485195											
485196											
485197											
485198											

- How many Swiss-Prot Identifiers are in the file ? ✓
- How many different GO annotations are there ? ✗
- What are the 4 most common GO annotations ? ✗

CLI vs GUI - complex queries

- *How many Swiss-Prot identifiers?*

```
userA$ wc -l go-data.csv
485194
```

- *How many different GO annotations?*

```
userA$ tr " " "\n" < go-data.csv | grep ^GO: | sort | uniq | wc
-l
15696
```

- *What are the 4 most common GO annotations?*

```
userA$ tr " " "\n" < go-data.csv | grep ^GO: | sort | uniq -c |
sort -nr | head -4
127303 GO:0005737
82740  GO:0005524
66591  GO:0016021
52952  GO:0046872
```

Outline

1. What is UNIX and why should biologists use it?
2. UNIX filesystem: navigation and usage
3. Environment, processes & Jobs
4. Working with directories and files
5. Working with file content
6. Optional exam

Exercises are integrated in chapters

UNIX filesystem: navigation and usage

Swiss Institute of Bioinformatics

Files & directories

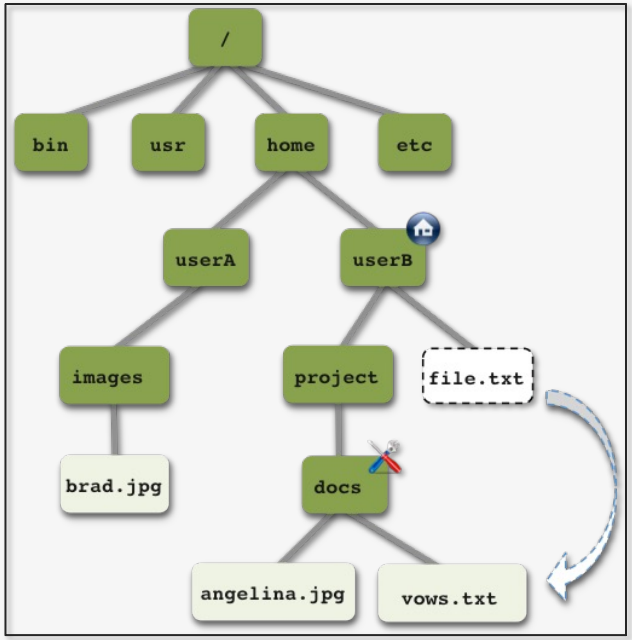
- Organised as an inverted tree
- Single root '/ ' at the top.

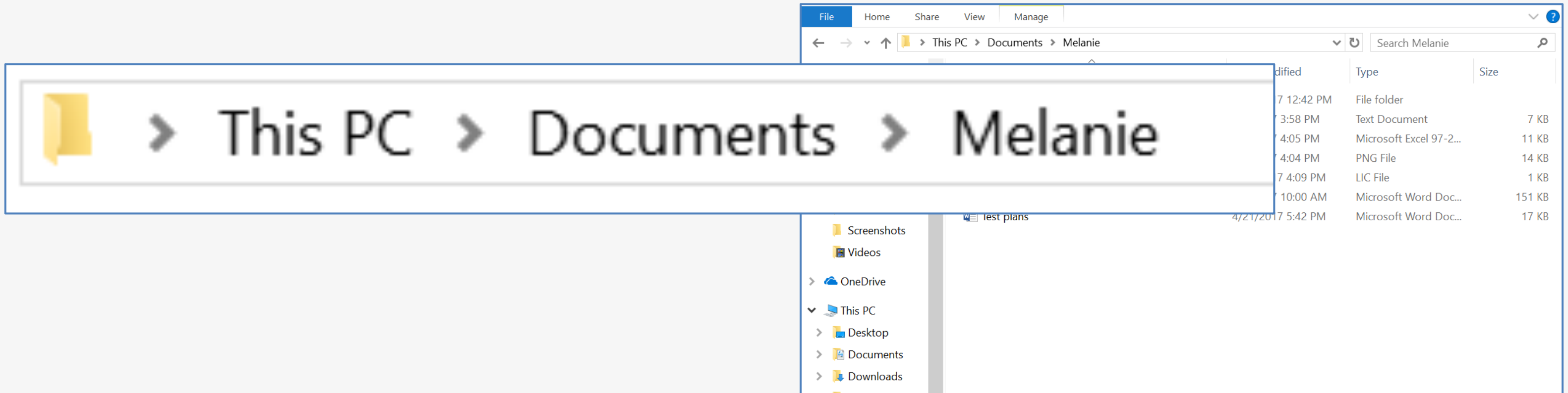
🟩 Directory: is a file containing file & directory names

🏠 Home directory: is the 'private' directory of a user

🛠 Working directory: is the current directory.
The one you are in when executing a command.

⬜ File: is a file



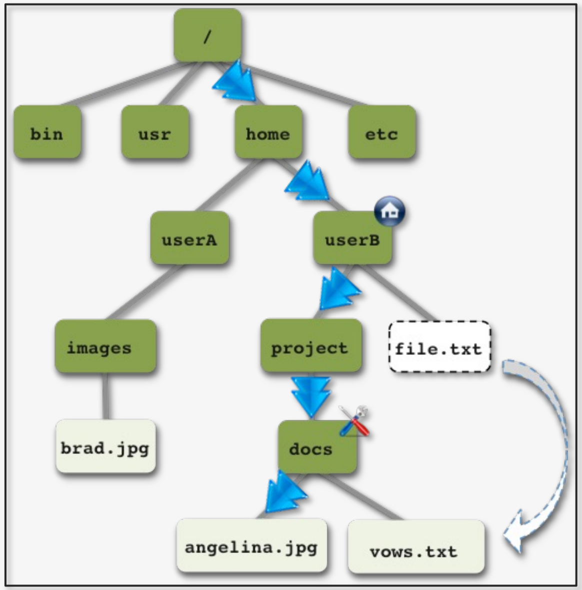


This PC › Documents › Melanie

	dified	Type	Size
	7 12:42 PM	File folder	
	3:58 PM	Text Document	7 KB
	4:05 PM	Microsoft Excel 97-2...	11 KB
	4:04 PM	PNG File	14 KB
	7 4:09 PM	LIC File	1 KB
	10:00 AM	Microsoft Word Doc...	151 KB
Test plans	4/21/2017 5:42 PM	Microsoft Word Doc...	17 KB

Paths

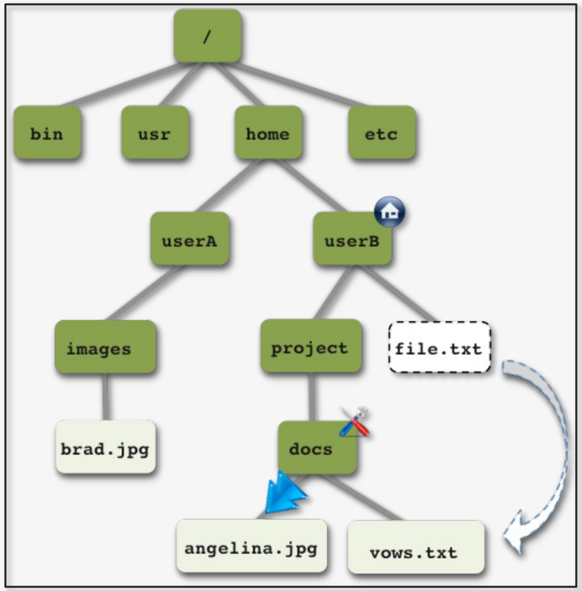
absolute path...



from the root

```
/home/userB/project/docs/angelina.jpg
```

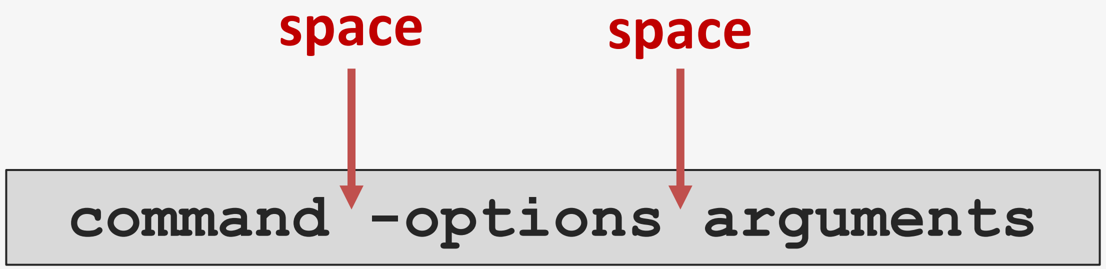
relative path...



from the working directory

```
angelina.jpg
```

General command syntax



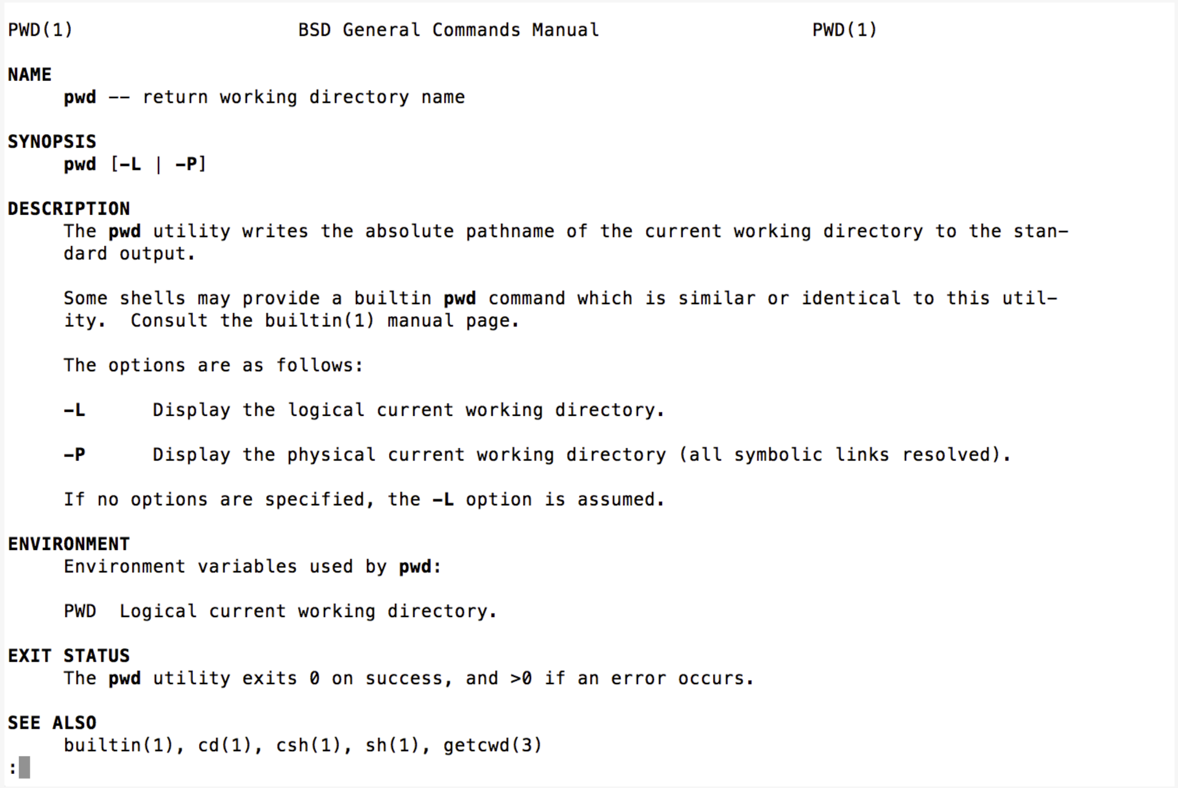
```
command -options arguments
```

`command`	**program** performing an action
`-option(s)`	modifies the command
`argument(s)`	identifies data (file or directory, with or without a path)

By default, commands are executed in the working directory

Help about commands & options

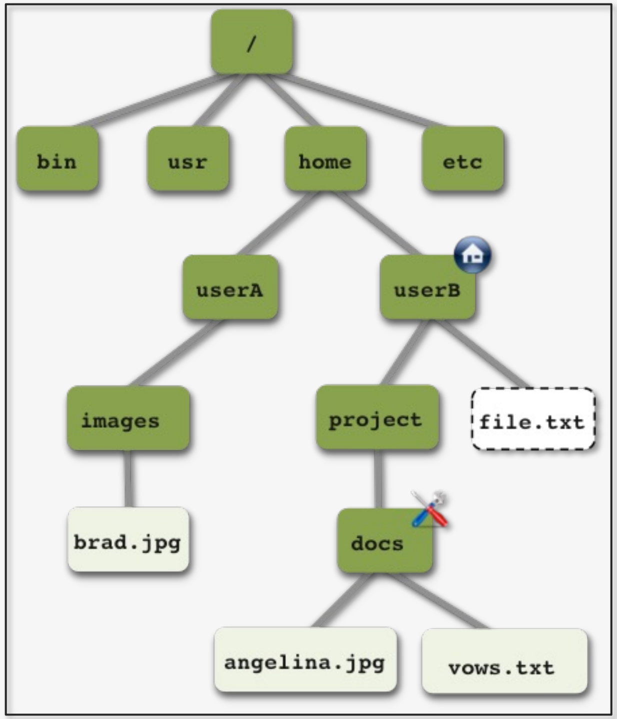
man



```
PWD(1)                    BSD General Commands Manual                   PWD(1)

NAME
     pwd -- return working directory name

SYNOPSIS
     pwd [-L | -P]

DESCRIPTION
     The pwd utility writes the absolute pathname of the current working directory to the stan-
     dard output.

     Some shells may provide a builtin pwd command which is similar or identical to this util-
     ity.  Consult the builtin(1) manual page.

     The options are as follows:

     -L      Display the logical current working directory.

     -P      Display the physical current working directory (all symbolic links resolved).

     If no options are specified, the -L option is assumed.

ENVIRONMENT
     Environment variables used by pwd:

     PWD  Logical current working directory.

EXIT STATUS
     The pwd utility exits 0 on success, and >0 if an error occurs.

SEE ALSO
     builtin(1), cd(1), csh(1), sh(1), getcwd(3)
:
```

Type q to exit

Navigation - Locate



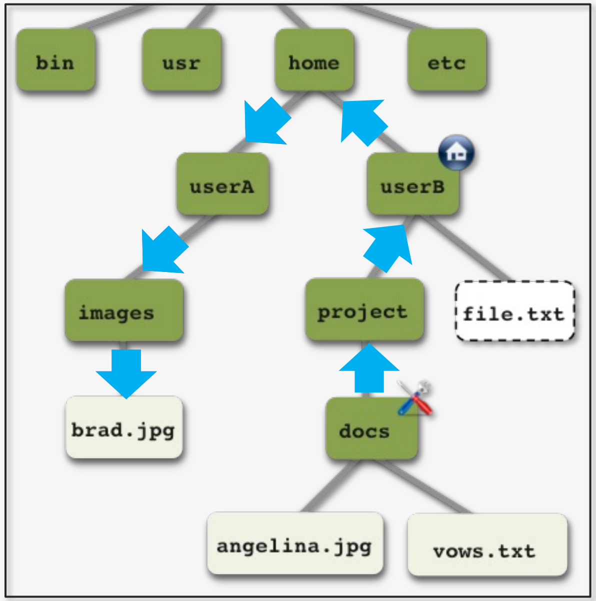
pwd

- for 'print working directory'
- indicates the absolute path of the current directory

```
userB$ pwd
/home/userB/project/docs
```

Current & parent directories

Example: the current directory is 'docs'



Absolute path to brad.jpg

```
/home/userA/images/brad.jpg
```

Relative path to brad.jpg

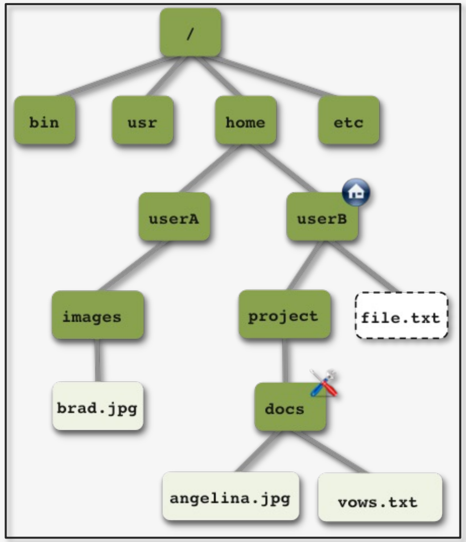
```
../../../userA/images/brad.jpg
```

Parent directory

Copy brad.jpg to current directory

```
../../../home/userA/images/brad.jpg .
```

Navigation



cd

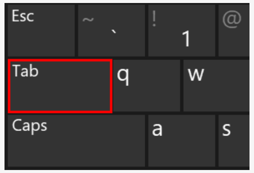
- for 'change directory'
- allows to navigate through the directories

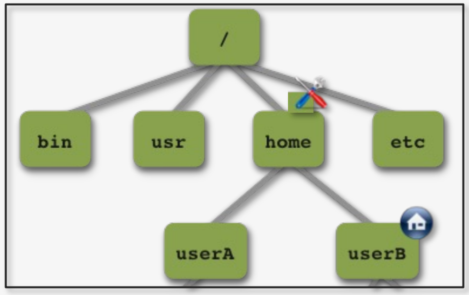
  - To go to the root: `cd /`
  - To go to user home directory: `cd ~` or just `cd`
  - To go up: `cd ..`
  - To go down: `cd project/docs`

Navigation – Auto-completion

Current working directory: home

[command] + 1x

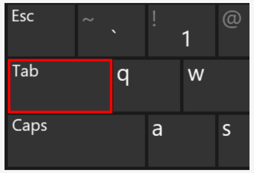




-> auto-completion until a ''choice''.

- Example:
  userA
  userB

- [command] + 2x



-> lists the content of the current directory

Navigation – Auto-completion

- Fast navigation
- Safe commands

Navigation – Content listing

ls

- for 'list'
- displays the directories and files of the current directory or of a specified directory

Options

- **-l** shows one item per line, with **detailed information**, such as dates and access rights
- **-h** prints file and directory sizes in **human readable format** (combined with -l)
- **-a** shows all files, including **hidden files**
- **-t** sorts by **modification time** (most recent first), default sorting being by name
- **-r** **reverts** sorting
- **-R** shows content recursively

Rules – case sensitivity

The command line differenciates **upper** from **lower** case letters:

brad.jpg

≠

BRAD.jpg

≠

brad.**JPG**

Rules – filename limitations

Nearly no limitation in length or characters. However…

- forbidden character: **/**
- not recommended characters:
  - characters with a particular meaning for the shell: **space , ; : * ? & % $ | ^ ~ ' " () [] {} ! \ #**
  - international characters: **é à æ ñ ç**
- safe characters: **- _ . 0-9 a-z**

Rules – filename extensions

File extensions are **arbitrary** and **do not have a particular meaning** for the operating system, but are useful for the users.

.sh	shell scripts
.pl	perl scripts
.py	python scripts
.txt	text files with no particular format
.csv	text files with **C**omma-**S**eparated **V**alues
.fas	files containing sequences in *FASTA* format*

**FASTA is a text-based format for representing either nucleotide or peptide sequences.*

Rules – filename extensions

A file can have several extensions…

archive.**tar**.**gz**

tar = archive
gz = compression

✓

archive.**od**.**cw**

od = no meaning
cw = no meaning

…or no extension

archive

Users and permissions

Every file / directory has access properties

who can access? 3 levels of classes:

user	the file owner
group	other users in the same group as the file owner
others	all other users in the system
all	all users in the system

Users and permissions

Every file / directory has access properties

what kind of access? 3 access modes:

read	permission to view the content of the file
write	permission to edit the content of the file
execute	permission to run a file (scripts, programs) or to enter a directory

Users and permissions

Every file / directory has access properties

what **restriction/permission**?

+	adds the permission
-	removes the permission
=	applies the specified mode to the specified classes (faster if heterogeneous initial modes)

Root or system administrator has all rights!!!

Permission modifications

chmod

```
chmod [u g o a] [+ - =] [r w x] file_name
```

▬	user	r	read
▬	group	w	write
▬	others	x	execute

Option:

-R recursive action

chmod can be run by root or by file owner only

Permission modifications

```
-rwxrwxrwx file01.txt
-rwxrwxrwx file02.txt
```

How to allow everyone to read these files, but prevent 'group' + 'others’ to 'write' and 'execute’ them?

```
chmod go-wx file01.txt file02.txt
```

```
-rwxr--r-- file01.txt
-rwxr--r-- file02.txt
```

Permission modifications

```
-rwxrwx--- file01.txt
-rwxr-xr-x file02.txt
```

How to allow everyone to read these files, but prevent 'group' + 'others' to 'write' and 'execute' them?

When several files with different permissions, it will be useful to set exact permissions using '='

```
chmod u=rwx,go=r-- file01.txt file02.txt
```

```
-rwxr--r-- file01.txt
-rwxr--r-- file02.txt
```

Permission modifications

Each configuration of permission can be replaced by a code.

For example ***chmod 700*** = all permissions for owner, but none for all others

ugo

Number	Octal Permission Representation	Ref
0	No permission	---
1	Execute permission	--x
2	Write permission	-w-
3	Execute and write permission: 1 (execute) + 2 (write) = 3	-wx
4	Read permission	r--
5	Read and execute permission: 4 (read) + 1 (execute) = 5	r-x
6	Read and write permission: 4 (read) + 2 (write) = 6	rw-
7	All permissions: 4 (read) + 2 (write) + 1 (execute) = 7	rwx

In a nutshell

- Files and directories are organised as a tree
  - Navigate along its branches with cd
  - List folders content with ls
- Commands are executed at the current working directory if a path is not specified
- Rules
  - Case sensitivity
  - Filename extensions are arbitrary
  - Characters with a meaning for the shell are not recommended
- Access permissions for reading, writing and execution

Outline

1. What is UNIX and why should biologists use it?
2. UNIX filesystem: navigation and usage
3. Environment, processes & Jobs
4. Working with directories and files
5. Working with file content
6. Optional exam

Exercises are integrated in chapters

Environment, processes & Jobs

Swiss Institute of Bioinformatics

User environment

man

Equivalent to help, often used to search commands’ options

uname

Prints information about the system (several options)

whoami, id, groups, who

Prints information of the user

User environment

clear

This command allows to clean the screen

history

Allows to view the history of previous commands, which are listed with an increasing number

Call the last command with: !!

Search a string with ctrl + r

Browse history with up and down arrows

View page by page history with history | less

Browse through the last commands with the UP and DOWN keys

Alias

Regularly used complex commands can be renamed for simplification

Example:

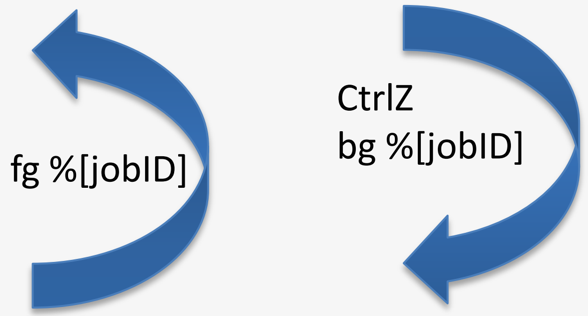
alias la="ls -altr"

(list the content, including hidden files, ordered by reverse date)

- active only during the current session
- write it in a special file to make it permanent

Job status

- Run in the foreground
  - **prompt not available** until the job ends
  - the **output** is **displayed** on the terminal
- Run in the background
  - add **&** after the command
  - the **prompt remains available**
  - the output is **not displayed** on the terminal
- Done
- Stopped
- Terminated / killed -> (CTRL -c, kill)
  - when a process does **not behave as expected** - frozen, gone into a loop...
  - Use **CTRL -c** when in foreground to kill the current process
  - Use **kill %[jobID]** to kill a job in background



In a nutshell

- Commands to get information about you and your environment
- History of the last previous commands
- Simplfy commands with alias
- Jobs can be running in foreground or background
- Kill a process / job with CTRL -c when you get stuck

Outline

1. What is UNIX and why should biologists use it?
2. UNIX filesystem: navigation and usage
3. Processes & Jobs
4. Working with directories and files
5. Working with file content
6. Optional exam

Exercises are integrated in chapters

Working with directories and files

Swiss Institute of Bioinformatics

Wildcard characters

are very useful for multiple files manipulation

***** matches **any number of characters** in a file or directory name

```
ls image*.jpeg   ls *.jp*g
```

? matches **exactly one character** - use '??' to match any group of 2 characters

```
ls *.???
```

[ ] specify a **range of characters** allowed at that position – separated by an hyphen; use '!' to exclude a range of characters at that position

```
ls [a-f]* ls [!a-f]*
```

{ } specify a **list of terms** - separated by commas

```
ls *{.html,.txt}
```

Directories & files organisation

mkdir

- "make directory": **creates** a new directory in the current directory
  mkdir new-directory(ies)

Option

-to create:

- several levels at a time: option –p
- several directories at a time with a path:

```
mkdir level1/{level2a,level2b,level2c}
```

- several level with several directories

```
mkdir -p l1/{L2a,L2b,L2c}
```

Directories & files organisation

mv

- **moves** a file or a directory to a new location

```
mv file-or-directory-name  new-location/
```

- **renames** a file or a directory

```
mv file-or-directory-name  new-name
```

Important: Moving files can overwrite existing files. -i option ask before overwritting

Directories & files organisation

```
mv file1 file2
```

If no file2, file1 is renamed
If file2 exists, it's overwritten -> use option –i for confirm.

```
mv file1 directory2/
```

file1 is moved to directory2

```
mv file1 directory2
```

file1 is renamed directory2

Good Practice: autocompletion is useful to check if the new name or the new location exist

Directories & files organisation

cp

- to **make a copy** of a file with another name
  ```
  cp file-name.txt file-name2.txt
  ```

- to **make a copy of** a file in another directory (same name is kept)
  ```
  cp file-name.txt Other_directory/
  ```

- to **copy a directory** and all its files
  ```
  cp -R directory/ new-directory/
  ```

Important: Copying files can overwrite existing files. Use option -i

Directories & files organisation

rm

- to remove/**delete a file**
  rm file-name.txt

rmdir

- to remove/**delete a directory**
  rmdir empty-directory-name/

Important: by default a directory must be empty before it can be deleted.

Directories & files organisation

rm options

- **-R** or **-r** (for 'recursive')
  removes directories and their contents recursively
  rm –R non–empty-directory-name/

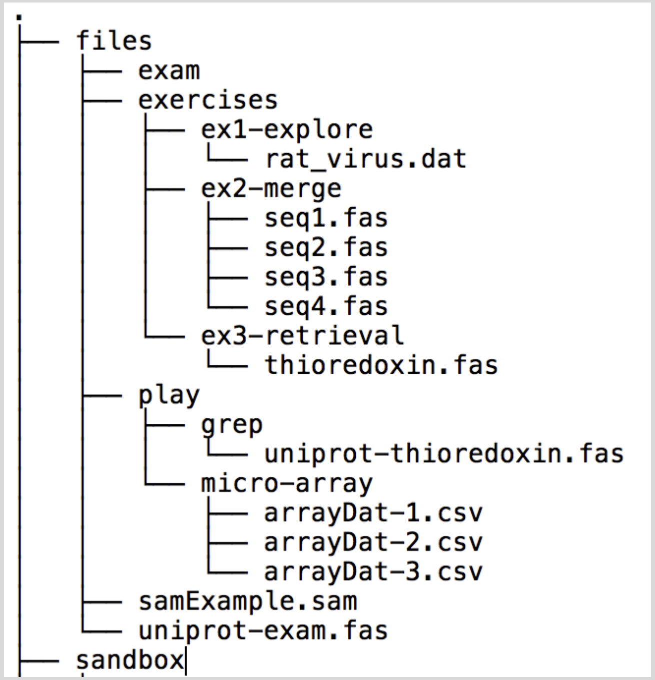
- **-f** (for '-force')
  never prompts for user confirmation, when files are write-protected

Use these options with care as there is no way back!

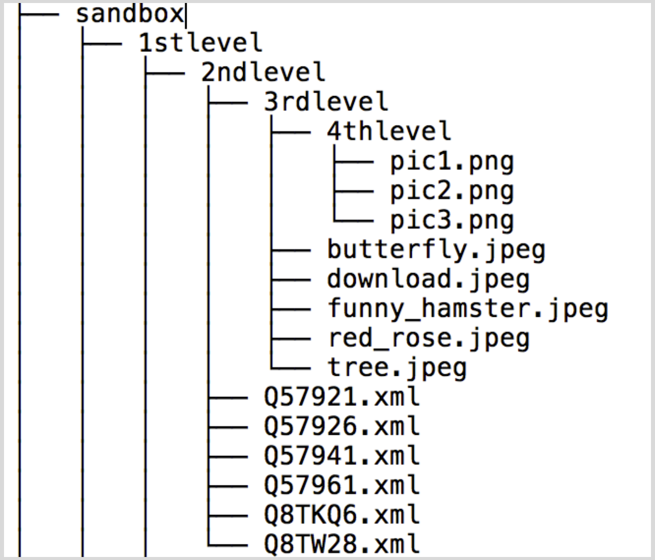
Hint: before executing a *rm* command with file names containing wildcard characters, simulate it first with the *ls* command as a control.

File organization

tree



```
.
├── files
│   ├── exam
│   ├── exercises
│   │   ├── ex1-explore
│   │   │   └── rat_virus.dat
│   │   ├── ex2-merge
│   │   │   ├── seq1.fas
│   │   │   ├── seq2.fas
│   │   │   ├── seq3.fas
│   │   │   └── seq4.fas
│   │   └── ex3-retrieval
│   │       └── thioredoxin.fas
│   ├── play
│   │   ├── grep
│   │   │   └── uniprot-thioredoxin.fas
│   │   └── micro-array
│   │       ├── arrayDat-1.csv
│   │       ├── arrayDat-2.csv
│   │       └── arrayDat-3.csv
│   ├── samExample.sam
│   └── uniprot-exam.fas
├── sandbox
```



```
├── sandbox
│   ├── 1stlevel
│   │   ├── 2ndlevel
│   │   │   ├── 3rdlevel
│   │   │   │   ├── 4thlevel
│   │   │   │   │   ├── pic1.png
│   │   │   │   │   ├── pic2.png
│   │   │   │   │   └── pic3.png
│   │   │   │   ├── butterfly.jpeg
│   │   │   │   ├── download.jpeg
│   │   │   │   ├── funny_hamster.jpeg
│   │   │   │   ├── red_rose.jpeg
│   │   │   │   └── tree.jpeg
│   │   │   ├── Q57921.xml
│   │   │   ├── Q57926.xml
│   │   │   ├── Q57941.xml
│   │   │   ├── Q57961.xml
│   │   │   ├── Q8TKQ6.xml
│   │   │   └── Q8TW28.xml
```

Find files or directories

find

- Very effective in finding files & directories
  find [path] [query type] 'query term'

Path

- '.' to search in the current directory and its sub-directories
- '/' (root directory) to search in all directories
- Use single or multiple '../' to search in parent directories
- '~' to search in your home directory

Find files or directories

Query types

- **-name** for a search based on the file name
  -iname same but case insensitive

```
find . -name '*.java'
```

- **-type f** or **-type d** to limit the search to files only or directories only

- **-size [+,-]n[scale]**, for a search based on the file size, where…
  + stands for 'larger than', **-** for 'smaller than', nothing for 'exact size',
  n indicates the numeric value,
  scale indicates the size: **c** for bytes; **k** for kilobytes; **M** for megabytes and **G** for gigabytes

```
find . -size +50M
```

- **-perm** [permission code]

```
find . -perm 644
```

Find files or directories

Tips

- Better to quote the query term with simple quotes
- Wildcards are possible
- ! placed after the path is the negative (*"that do no match the criteria"*)
- Options can be combined
- Boolean operators (AND, OR) can be used in combination with find

```
find . -name '*.mpeg' -and -size +30M (default)
find . -name '*.png' -or -name '*.jpg'
find . \( -name '*.png' -or -name '*.jpg′ \)
-and -size +30M
```

When multiple arguments, possible to use regular expressions (-regex option)

File type determination

Reminder: file extensions **do not have a particular meaning** for the OS, thus might not correspond to the real file format

file

To check the **format** and detailed information of a file:

file filename(s)

- **text** - the file contains only printing characters and a few common control characters
- **executable** - the file contains the result of compiling a program
- **data** - anything else (data is usually binary or non-printable)

New file

touch

- creates a **new** empty file
  touch filename(s)

If *filename* already exists, this command does not erase file content, but updates its latest modification time

Archiving and extracting data

tar

- Creates, updates or expands archives

Options

-z	compresses using gzip program
-c	creates an archive
-x	extracts from an archive
-t	displays archive content (but does not extract it)
-v	displays information on the terminal window
-f	uses specified file
-m	set the current dates to extracted files

type of action

Archiving and extracting data

Highly recommended filename extensions:

- **.tar** for all archives
- **.gz** for gzip compressed archives

create an archive: `tar -zcvf archive.tar.gz myfiles`

show archive content: `tar -ztvf archive.tar.gz`

extract data from archive: `tar -zxvf archive.tar.gz`

extract to another location: `tar -zxvf archive.tar.gz -C location/…`

set the current date: `tar -zxvmf archive.tar.gz`

upate an uncompressed archive `tar -uf archive.tar new-files`

Tip: -f option must be placed at the last position because it requires argument (file name)

In a nutshell

- Directory and file manipulation
  - mkdir, mv, cp, rm, rmdir
- Finding files and directories (find)
  - Wildcards characters for file search and display
- File determination (file)
- Archiving/extracting files (tar)

Outline

1. What is UNIX and why should biologists use it?
2. UNIX filesystem: navigation and usage
3. Processes & Jobs
4. Working with directories and files
5. **Working with file content**
6. Optional exam

Exercises are integrated in chapters

Working with file content

Line by line

Rule:

Content search is performed

line by line

File content statistics

wc

wc option filename(s)

Options

• **-l**	prints the number of lines
• **-w**	prints the number of words
• **-c**	prints the number of bytes
• **-m**	prints the number of characters

Without options, **wc** outputs the number of lines, words and bytes

File content display: head & tail

head

- aims at quickly showing the **beginning of a file**

head *myfile.txt* -> displays the first 10 lines (default)

head *-2 myfile.txt* -> displays the first 2 lines

head *-n-2 myfile.txt* -> displays all the lines, except the last 2

tail

- is similar to **head**, but shows the **last lines** of a file

tail *myfile.txt* -> displays the last 10 lines

tail *-2 myfile.txt* -> displays the last 2 lines

tail *-n+2 myfile.txt* -> displays all lines except the first (useful e.g. to remove header line)

File content display: cat & less

cat displays the content of a text file

When several files are passed as arguments, **cat** concatenates and displays their content, thus can be used to merge the content of several files

File content display: cat & less

less

- displays the content of a text file **one screen at a time**

Option

-N shows line numbers at the left of each line

Useful commands to be used inside the text file:

- **[space-bar]**, shows the next page, one screen at a time
- **up/down arrow keys**, moves the content backward/forward one line at a time
- **<** and **>** to reach the beginning and the end of the document
- **/mouse**: search the string “mouse”
- **q** to quit

Text editors

Interactive use –> no need for text edition

Programmning -> need for text edition

Several programs available: Sublime Text, Visual Studio Code, etc

Vim is a text editor integrated into most terminals

Nano

Standard streams

UNIX has three standard streams

- **Standard input** - **stdin**, where programs receive data from:
  - from the **keyboard**
  - from a **file** (using **<**)

- **Standard output** - **stdout**, where programs write their output data to:
  - **printed** on the terminal window
  - **written** in a file (using **>** or **>>**)

- **Standard error** - **stderr**, to output error messages
  - **printed** on the terminal window

Standard streams

Example of **standard input** from file

```
command < ID_B.txt
```

< not mandatory with most programs

Standard streams

Example of standard **output to a file**

Command **>** *filename*

- if the file does not exist, it is created
- If it does, its content is **overwritten**

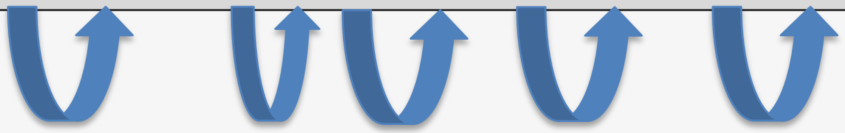
Command **>>** *filename*

- if the file does not exist, it is created (same function as **>**).
- If it does, the output is **added at the end** of the file content.

Standard streams

Output redirection: the | (pipe):

```
userA$ tr " ” "\n" < go-data.csv | grep ^GO: | sort | uniq -c | sort -nr | head -4
```



useful to pass directly the output from one program as the input of another **without creating intermediate files**

Delimiter-Separated Values (DSV)

Free-text (not DSV)

```
Hello, I will be very busy the next couple of months: I started a
course in Bioinformatics!
```

Comma/Semicolon-separated values (CSV)

```
Entry,Entry_name,Protein_names

Q8QMT5,A18_CWPXB,Transcript termination protein A18
Q80DV6,A18_CWPXG,Transcript termination protein A18
```

Tab-separated values (TSV)

```
Entry                 Entry name          Protein name

Q8QMT5                A18_CWPXB           Transcript termination protein
A18 Q80DV6                      A18_CWPXG           Transcript termination
protein A18
```

text identification, retrieval and manipulation is more efficient when structured with delimiters

Delimiter-Separated Values (DSV)

Biological sequences are often represented in a so-called **FASTA** format:

- a first line starting with '>' and free text / comments (header line)
- main lines composed of one-letter code sequences

```
>sp|Q1PE49|4ON1_ARATH Protein At-4/1 OS=Arabidopsis thaliana
MAATSDEQMNLLLSSFDQIYEDFKIGLNEINVYRSKSNVESSRREVLEISNKNLKEENER
LKKLYTESLNNFADQLEHRTKCHSLKEELKRVNDENKSKEHEHRNALESLRQKHVTKVEE
```

Newline character

structured formats → a 'newline' character* at the end of each line, which **IS NOT** the same for all OS

- **\n** -> Unix, Mac OS X
- **\r** -> Mac OS (before X)
- **\r\n** -> Windows OS

*use command *od -c* to view them

conversion is sometimes necessary

Converting file content

To convert newline characters…

tr (for 'translate') can do this in a single command line:

```
tr '\r' '\n' < myfile  > mynewfile
```

tr is very convenient to change a delimiter, e.g. from TSV to CSV

If you want to replace words or phrases, the command **sed** is required (not discussed during this course)

Sorting file content

sort

- used to sort lines of text files
- default is alphabetical order

Options

- **-n** for a numeric sort
- **-r** for reverse sort
- **-u** for unique lines (keeps one instance when identical lines)
- **-k** sorts via a key (a column number for example)
- **-t** to define a key separator (e.g. a |). Default is space, but tab is recognized too

sort -k is useful if you want to sort according to only some parts of a line, typically columns

Searching / filtering file content

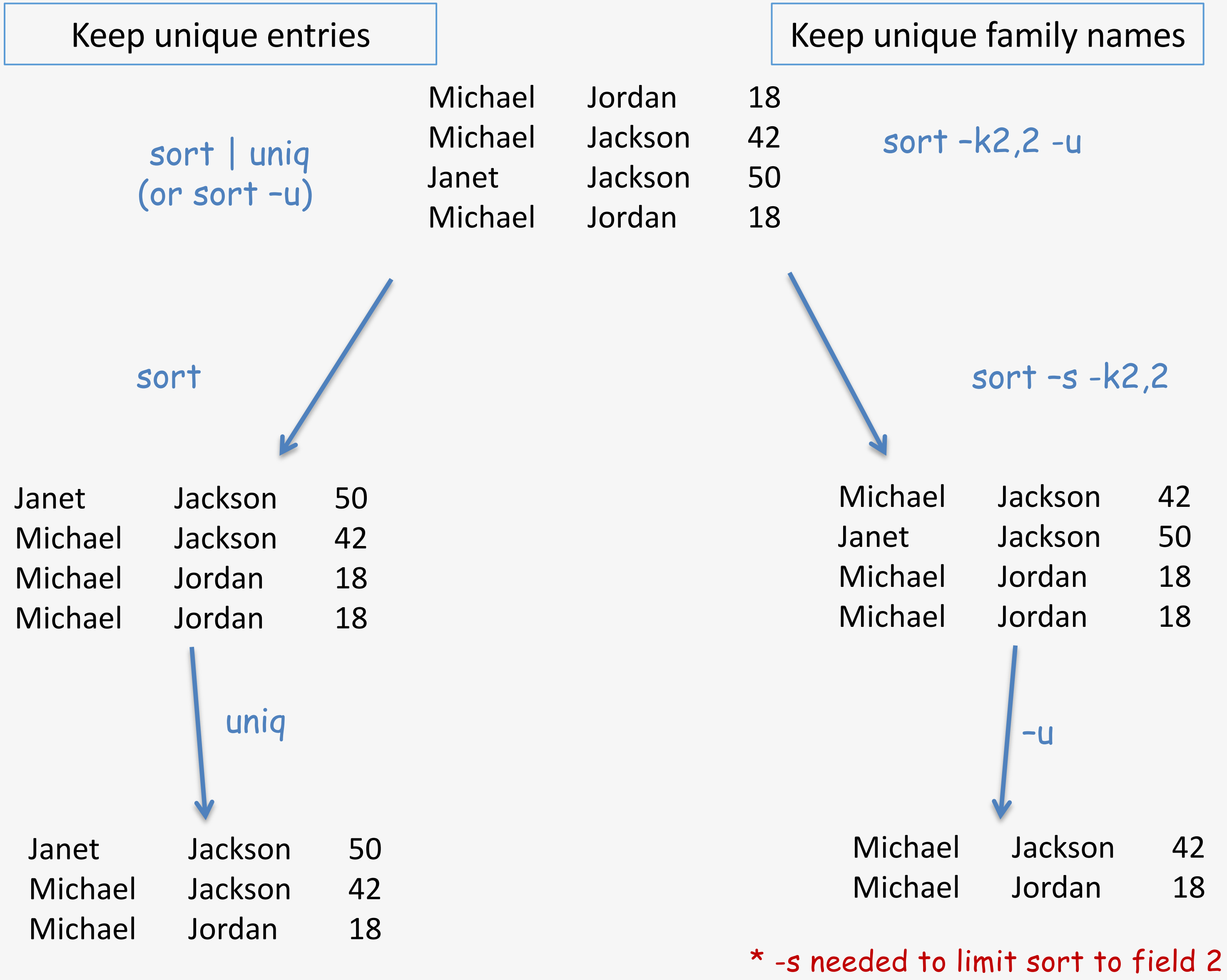
uniq

- reports **unique lines** by suppressing multiple occurrences of **identical** lines if **sequential**. Therefore, it is typically used after **sort**

Options

- **-d**, outputs only lines that are repeated (duplicated or more) in the input
- **-c**, precedes each reported line with the **number of occurrences** found
- **-i**, ignores the differences in case when comparing the lines
- **-u**, only prints unique lines

sort | uniq and sort -u have the same function: suppression of multiple occurrence of identical lines and ordering, but sort has an option (-k) to sort on specific key (e.g. a column) instead of the entire line.


Keep unique entries

Keep unique family names

```
Michael     Jordan      18
Michael     Jackson     42
Janet       Jackson     50
Michael     Jordan      18
```

```
sort | uniq
(or sort -u)
```

```
sort
```

```
Janet       Jackson     50
Michael     Jackson     42
Michael     Jordan      18
Michael     Jordan      18
```

```
uniq
```

```
Janet       Jackson     50
Michael     Jackson     42
Michael     Jordan      18
```

```
sort -k2,2 -u
```

```
sort -s -k2,2
```

```
Michael     Jackson     42
Janet       Jackson     50
Michael     Jordan      18
Michael     Jordan      18
```

```
-u
```

```
Michael     Jackson     42
Michael     Jordan      18
```

* -s needed to limit sort to field 2

Filtering / extracting file content

cut is used to extract selected parts of lines (like columns).
Convenient to manipulate delimiter-separated value files

Options

- **-c**, selection based on characters positions. This is used when cut has to be performed at a fixed position (e.g. to remove a fixed header)
- **-d**, type of delimiter (default is 'tab'), e.g. ';' ',' ' ' or any character
- **-f**, selection based on fields in DSV files.
  single field number (e.g. *-f 1*) , multiple field numbers (e.g. *-f 1,4*) or a range (e.g. *-f 1-4*) must be indicated. Fields are determined by delimiters if not TAB

NB: `cut -f 1 | sort -u` and `sort -k1,1 -u` are almost identical but the former will display only the extracted column

Comparing file content

diff compares **2** files and outputs the differences.

Useful to compare:

- 2 files with the same name in different directories
- different versions of the same file

Options

- **-b**, ignores differences in amount of white space
- **-B**, ignore differences of blank lines
- **-s**, reports when two files are identical (it does not by default)
- **-q**, reports only whether two files differ, no details of the differences

Merging file content

cat

- display function -> can be used to concatenate content from several files
- concatenation does not include any blank or newline

Options

- **-s** , squeezes multiple adjacent empty output lines
- **-b**, numbers nonempty output lines

paste

- can merge columns from different files

Options

-d allows to introduce delimiters between columns (':' ',' '-')

Searching file content

grep

- for **G**lobal **R**egular **E**xpression **P**rint - searches file(s) for words or patterns and return matching lines
  grep [options] 'pattern' file(s)

Options

- **-c**, displays the **number** of resulting lines instead of the lines themselves
- **-i**, performs a **case insensitive** search
- **-v**, displays lines that **DO NOT** match the pattern
- **-o**, returns only the pattern, not the entire line
- **-n**, adds the **line number** in front of the result lines
- **-r**, search all files **recursively** inside each directory

Regular Expressions

Characters used to search a pattern, example:

^a	‘a’ at the beginning of a line
a$	‘a’ at the end of a line
^$	to express a blank line
^.$	any character on a line
[abc]	either character
[a-z]	a lowercase character
[0-9]	any figure
[0-59]	0, 1, 2, 3, 4, 5 or 9
[^0-9]	any character except a figure

http://www.grymoire.com/Unix/Regular.html

In a nutshell

- UNIX allows to manipulate file content, such as order lines, cut fields, merge file content
- UNIX allows complex searches to retrieve filtered data
- Complex commands might be written into scripts
- Note that other languages are better suited to this sort of complex manipulations, like AWK, perl, python, or R

Next step

UNIX shell scripting in Life Sciences

By Thomas Junier & Robin Engler

13-15 June 2023

Lausanne

https://www.sib.swiss/training/course/20230606_ADVUN

Further resources

Our e-learning tutorial:

https://edu.sib.swiss/course/view.php?id=82

Resources that could be of interest:

http://en.wikipedia.org/wiki/List_of_Unix_commands
http://www.catonmat.net/projects/cheat-sheets/
https://ubuntudanmark.dk/filer/fwunixref.pdf

Acknowledgements

- Diana Marek
- Vassilios Ioannidis
- Volker Flegel
- Frédéric Schütz
- Heinz Stockinger
- Ivan Topolsky
- Alex Smith

Thank you for your attention!

http://sib.swiss/training

training@sib.swiss

20 YEARS SIB Swiss Institute of Bioinformatics